In [1]:
```
import os
import pandas as pd                                     # panda's nickname is pd
import numpy as np                                      # numpy as np
from pandas import DataFrame, Series                    # for convenience
import matplotlib.pyplot as plt
from datetime import datetime
from datetime import date
import time
import plotly.plotly as ply
import json
from fbprophet import Prophet


import tensorflow as tf
from tensorflow.contrib.timeseries import NumpyReader
# from tensorflow.contrib.timeseries.python.timeseries import NumpyReader
tf.logging.set_verbosity(tf.logging.WARN)
tf.__version__

%matplotlib inline
import matplotlib
# import seaborn as sns
# sns.set()
# sns.set_context("notebook", font_scale=1.2, rc={"lines.linewidth": 1.5})
```

In [2]:
```
divvy_trips = pd.read_csv('Divvy_Trips_2018.csv')
```

In [3]:
```
divvy_trips.head()
```

Out[3]:

	trip_id	start_time	end_time	bikeid	tripduration	from_station_id	from_station_name	to_station_id	to_station_name	usertype	gender	bi
0	19244622	7/1/2018 0:00	7/1/2018 23:56	5429	86,168.00	140	Dearborn Pkwy & Delaware Pl	106	State St & Pearson St	Customer	NaN	
1	19244623	7/1/2018 0:00	7/1/2018 0:06	93	386	153	Southport Ave & Wellington Ave	250	Ashland Ave & Wellington Ave	Subscriber	Male	
2	19244624	7/1/2018 0:00	7/1/2018 0:23	2461	1,391.00	76	Lake Shore Dr & Monroe St	301	Clark St & Schiller St	Subscriber	Female	
3	19244625	7/1/2018 0:00	7/1/2018 0:23	2991	1,386.00	76	Lake Shore Dr & Monroe St	301	Clark St & Schiller St	Subscriber	Male	
4	19244626	7/1/2018 0:00	7/1/2018 0:11	2851	656	60	Dayton St & North Ave	166	Ashland Ave & Wrightwood Ave	Subscriber	Male	

In [4]:
```
divvy_trips.columns
```

Out[4]:
```
Index(['trip_id', 'start_time', 'end_time', 'bikeid', 'tripduration',
       'from_station_id', 'from_station_name', 'to_station_id',
       'to_station_name', 'usertype', 'gender', 'birthyear'],
      dtype='object')
```

In [5]:
```
divvy_trips['trip_id'] = divvy_trips['trip_id'].apply(str)
```

In [6]:
```
divvy_trips['start_time'] = pd.to_datetime(divvy_trips['start_time'])
```

In [7]:
```
divvy_trips['start_time'].head()
```

Out[7]:
```
0   2018-07-01
1   2018-07-01
2   2018-07-01
3   2018-07-01
4   2018-07-01
Name: start_time, dtype: datetime64[ns]
```

In [8]:
```
divvy_trips['end_time'] = pd.to_datetime(divvy_trips['end_time'])
```

In [9]:
```
divvy_trips.describe()
```

Out[9]:

	bikeid	from_station_id	to_station_id	birthyear
count	1.048575e+06	1.048575e+06	1.048575e+06	830155.000000
mean	3.427708e+03	1.887713e+02	1.900125e+02	1983.480110
std	1.926175e+03	1.386744e+02	1.385620e+02	10.698525
min	1.000000e+00	2.000000e+00	2.000000e+00	1895.000000
25%	1.726000e+03	7.700000e+01	7.700000e+01	1979.000000
50%	3.536000e+03	1.680000e+02	1.720000e+02	1987.000000
75%	5.136000e+03	2.830000e+02	2.840000e+02	1991.000000
max	6.471000e+03	6.310000e+02	6.310000e+02	2005.000000

In [10]:
```
divvy_trips['only_date'] = divvy_trips['start_time'].dt.date
```

In [11]:
```
divvy_trips['month'] = divvy_trips['start_time'].dt.month
```

In [12]:
```
divvy_trips['Week_number'] = divvy_trips['start_time'].dt.week
```

In [13]:
```
divvy_trips['hour'] = divvy_trips['start_time'].dt.hour
```

In [14]:
```
divvy_trips['day_of_week'] = divvy_trips['start_time'].dt.weekday_name
```

In [15]:
```
divvy_trips.dtypes
```

Out[15]:
```
trip_id                    object
start_time         datetime64[ns]
end_time           datetime64[ns]
bikeid                      int64
tripduration               object
from_station_id             int64
from_station_name          object
to_station_id               int64
to_station_name            object
usertype                   object
gender                     object
birthyear                 float64
only_date                  object
month                       int64
Week_number                 int64
hour                        int64
day_of_week                object
dtype: object
```

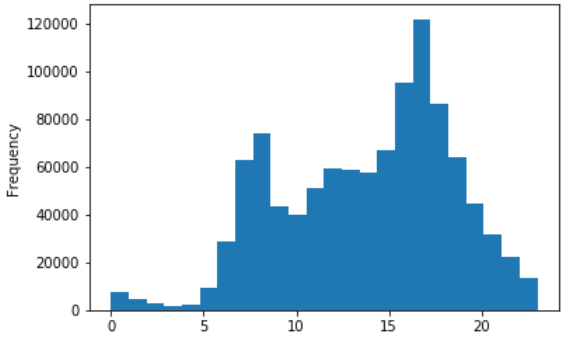
Q4 and Q5

In [16]:
```
divvy_trips['hour'].value_counts()
```

Out[16]:
```
17    121695
16     95012
18     86336
8      73788
15     66837
19     64099
7      62600
12     59447
13     58596
14     57413
11     51194
20     44330
9      43335
10     40081
21     31442
6      28769
22     21961
23     13277
5       9310
0       7760
1       4507
2       2900
4       2064
3       1822
Name: hour, dtype: int64
```

In [17]:
```
divvy_trips['hour'].plot(kind = 'hist', bins=24)
```

Out[17]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1dcc83c79e8>
```



Q6

In [18]:
```
divvy_trips['day_of_week'].value_counts()
```

Out[18]:
```
Wednesday    161748
Monday       155474
Tuesday      155408
Thursday     153417
Friday       142620
Saturday     140369
Sunday       139539
Name: day_of_week, dtype: int64
```

Q7

In [19]:
```
divvy_trips['Week_number'] .value_counts()
```

Out[19]:
```
30    137421
31    129352
33    124794
32    124663
27    123449
28    119926
29    115003
34    111419
35     50896
26     11652
Name: Week_number, dtype: int64
```

Q8

In [20]:
```
divvy_trips['month'].value_counts()
```

Out[20]:
```
7    544703
8    503872
Name: month, dtype: int64
```

Q9

In [21]:
```
divvy_trips['from_station_name'].value_counts()
```

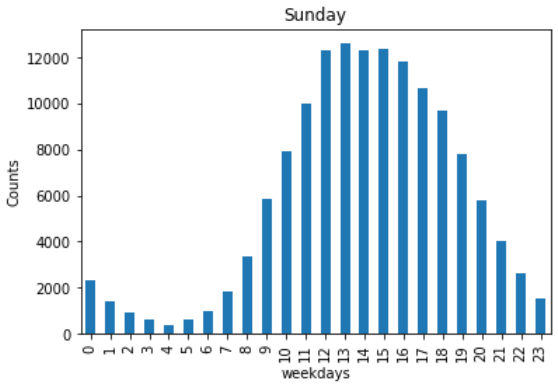
Out[21]:
```
Streeter Dr & Grand Ave                 27858
Canal St & Adams St                     17091
Clinton St & Madison St                 14569
Lake Shore Dr & North Blvd              14269
Lake Shore Dr & Monroe St               13419
Theater on the Lake                     13073
Michigan Ave & Oak St                   11833
Millennium Park                         11713
Michigan Ave & Washington St            11194
Clinton St & Washington Blvd            11083
Columbus Dr & Randolph St                9527
Daley Center Plaza                       9500
Franklin St & Monroe St                  9169
Kingsbury St & Kinzie St                 8855
Canal St & Madison St                    8750
Orleans St & Merchandise Mart Plaza      8270
Shedd Aquarium                           7565
LaSalle St & Jackson Blvd                7405
Clark St & Elm St                        7254
Dearborn St & Erie St                    6625
Wells St & Concord Ln                    6545
Fairbanks Ct & Grand Ave                 6489
Clark St & Armitage Ave                  6378
Wabash Ave & Roosevelt Rd                6125
Indiana Ave & Roosevelt Rd               6111
Wabash Ave & Grand Ave                   6110
Montrose Harbor                          6027
Clark St & Lincoln Ave                   5976
Michigan Ave & Lake St                   5891
McClurg Ct & Illinois St                 5845
                                        ...
Normal Ave & 72nd St                       16
Halsted St & 51st St                       16
Cicero Ave & Lake St                       16
Cicero Ave & Flournoy St                   16
Central Ave & Chicago Ave                  16
Greenwood Ave & 79th St                    15
Laramie Ave & Kinzie St                    15
Calumet Ave & 71st St                      15
Damen Ave & 51st St                        14
Central Ave & Madison St                   13
Stony Island Ave & South Chicago Ave       13
Ellis Ave & 83rd St                        13
Ashland Ave & Garfield Blvd                12
Central Park Ave & 24th St                 12
Central Park Blvd & 5th Ave                12
Cicero Ave & Quincy St                     11
Woodlawn Ave & 75th St                     11
Racine Ave & 65th St                       10
Shields Ave & 43rd St                      10
Damen Ave & 59th St                        10
DIVVY CASSETTE REPAIR MOBILE STATION       10
Ashland Ave & 66th St                       9
Laramie Ave & Gladys Ave                    9
Kostner Ave & Lake St                       8
Marshfield Ave & 59th St                    8
Seeley Ave & Garfield Blvd                  7
Laramie Ave & Madison St                    5
Kenton Ave & Madison St                     4
Racine Ave & 61st St                        3
Marion St & South Blvd                      1
Name: from_station_name, Length: 577, dtype: int64
```

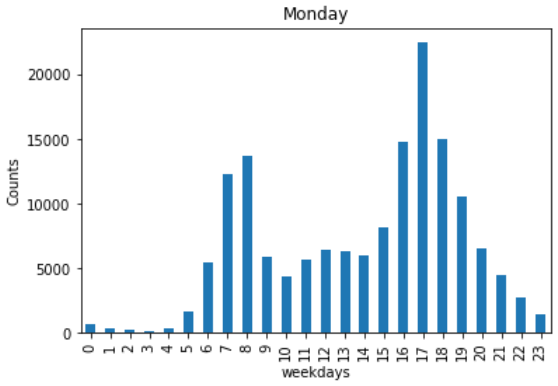
In [22]:

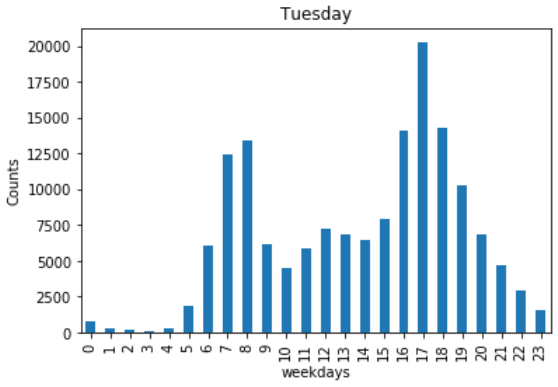
```
def plot_chart(divvy_trips):
    for i in ['Sunday', 'Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday' ]:

        plot_data = divvy_trips[divvy_trips['day_of_week'] == i]
        plot_data = plot_data.groupby('hour')['trip_id'].count()
        plot_data.plot(kind ='bar')
        plt.xlabel('weekdays')
        plt.ylabel('Counts')
        plt.title(i)
        plt.show()
```

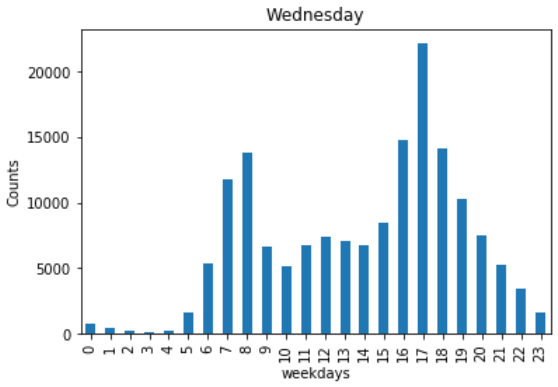
Q3

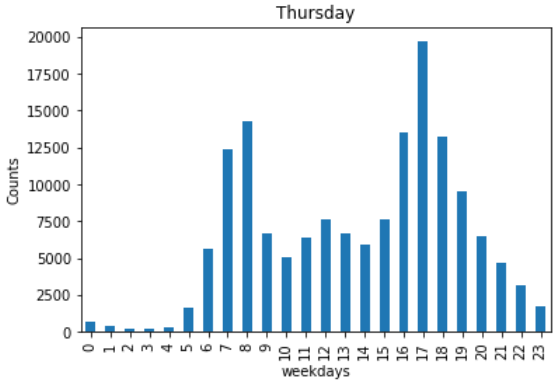
In [23]:
```
plot_chart(divvy_trips)
```

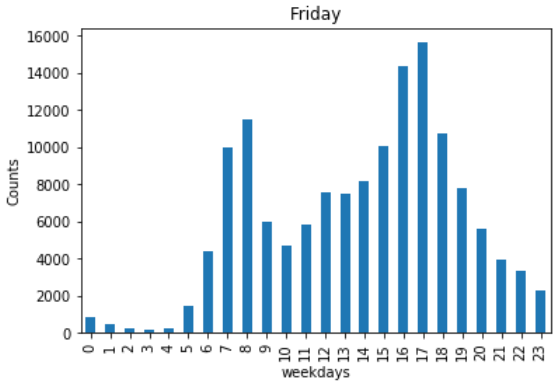


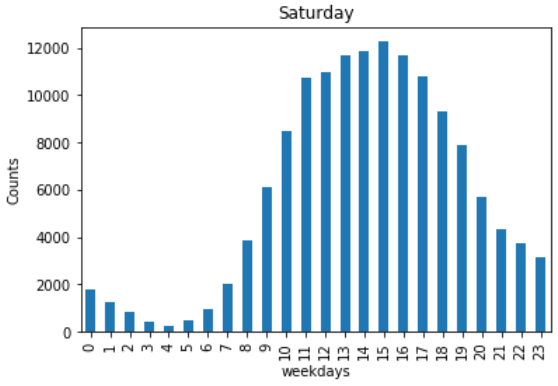












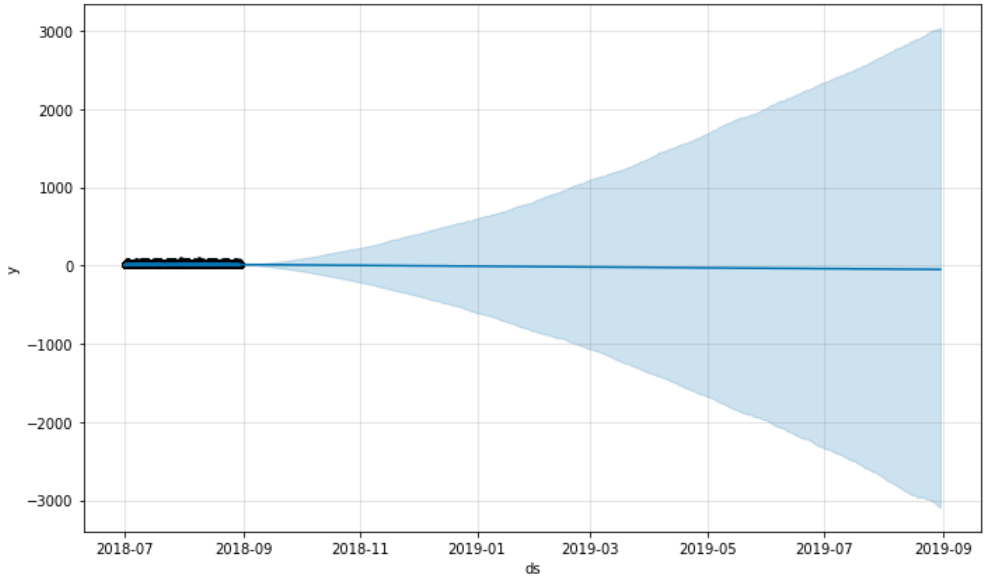
FB prophetModel

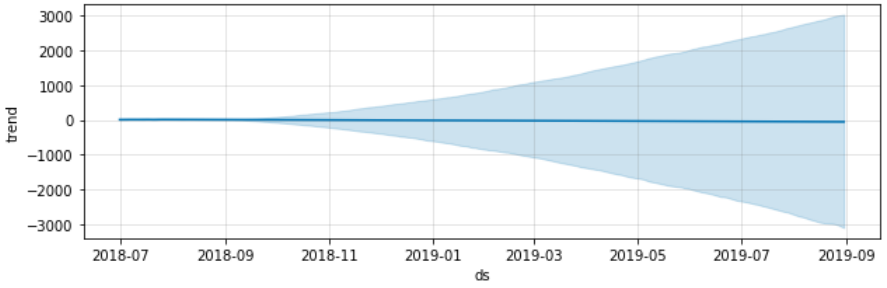
In [24]:
```
def Prophet_ModelForecast(input_df):
    model = Prophet()
    model.fit(input_df)
    future_df = model.make_future_dataframe(periods=365)
    future_df.tail()
    forecast = model.predict(future_df)

    forecast[['ds', 'yhat', 'yhat_lower', 'yhat_upper']].tail()
    plt.figure(figsize=(15, 8), dpi=80)
    fig1 = model.plot(forecast)
    fig2 = model.plot_components(forecast)
```

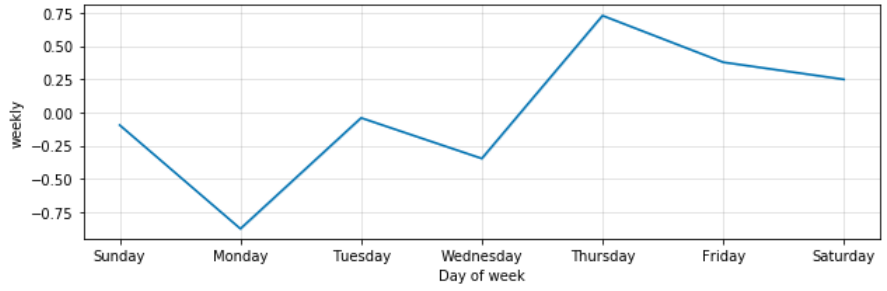
In [25]:
```
DF_Createdata = divvy_trips['start_time'].value_counts().rename_axis('ds').reset_index(name='y').sort_index()
```

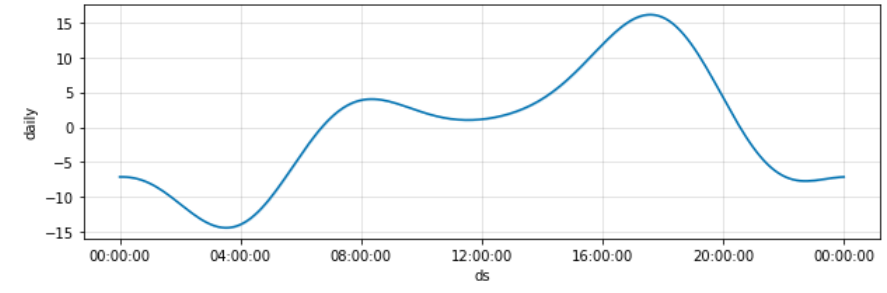
In [26]:
```
Prophet_ModelForecast(DF_Createdata)
```

```
C:\Users\chaet\Anaconda3\lib\site-packages\fbprophet\forecaster.py:880: FutureWarning:

Series.nonzero() is deprecated and will be removed in a future version.Use Series.to_numpy().nonzero() instead

INFO:fbprophet:Disabling yearly seasonality. Run prophet with yearly_seasonality=True to override this.
C:\Users\chaet\Anaconda3\lib\site-packages\pystan\misc.py:399: FutureWarning:

Conversion of the second argument of issubdtype from `float` to `np.floating` is deprecated. In future, it will be tr
eated as `np.float64 == np.dtype(float).type`.


<Figure size 1200x640 with 0 Axes>
```









Tensorflow model

In [27]:
```
def Tensorflowxy(x,y):
    print(x, y)
    data = {
        tf.contrib.timeseries.TrainEvalFeatures.TIMES: x,
        tf.contrib.timeseries.TrainEvalFeatures.VALUES: y
    }
    reader = NumpyReader(data)
    train_input_fn = tf.contrib.timeseries.RandomWindowInputFn(
            reader, batch_size=30, window_size=10)

    ar = tf.contrib.timeseries.ARRegressor(
            periodicities=200, input_window_size=5, output_window_size=5,
            num_features=1,
            loss=tf.contrib.timeseries.ARModel.NORMAL_LIKELIHOOD_LOSS)

    _ = ar.train(input_fn=train_input_fn, steps=600)
    evaluation_input_fn = tf.contrib.timeseries.WholeDatasetInputFn(reader)
    evaluation = ar.evaluate(input_fn=evaluation_input_fn, steps=100)
    (predictions,) = tuple(ar.predict(
            input_fn=tf.contrib.timeseries.predict_continuation_input_fn(
    evaluation, steps=10)))
    plt.figure(figsize=(12,6))
    x = data['times'].reshape(-1)
    y = data['values'].reshape(-1)
    plt.plot(x, data['values'].reshape(-1), ':', label='origin')

    x = evaluation['times'].reshape(-1)
    y = evaluation['mean'].reshape(-1)
    s = np.sqrt(evaluation['covariance'].reshape(-1))
    plt.plot(x, evaluation['mean'].reshape(-1), label='evaluation', color='black')
    plt.fill_between(x, y-1*s,y+1*s, alpha=0.1, color='blue')
    plt.fill_between(x, y-1*s,y+1*s, alpha=0.1, color='blue')

    x = predictions['times'].reshape(-1)
    y = predictions['mean'].reshape(-1)
    s = np.sqrt(predictions['covariance'].reshape(-1))
    plt.plot(x, y, label='prediction',color='orange')
    plt.fill_between(x, y-1*s,y+1*s, alpha=0.1, color='orange')
    plt.fill_between(x, y-1*s,y+1*s, alpha=0.1, color='orange')

    plt.xlabel('time_step')
    plt.ylabel('values')
    _ = plt.legend()
```

In [28]:
```
y = np.array(divvy_trips.groupby('only_date')['trip_id'].count())
```

In [29]:
```
x = np.array(range(len(y)))
```

In [30]:
```
x = pd.to_numeric(x)
```

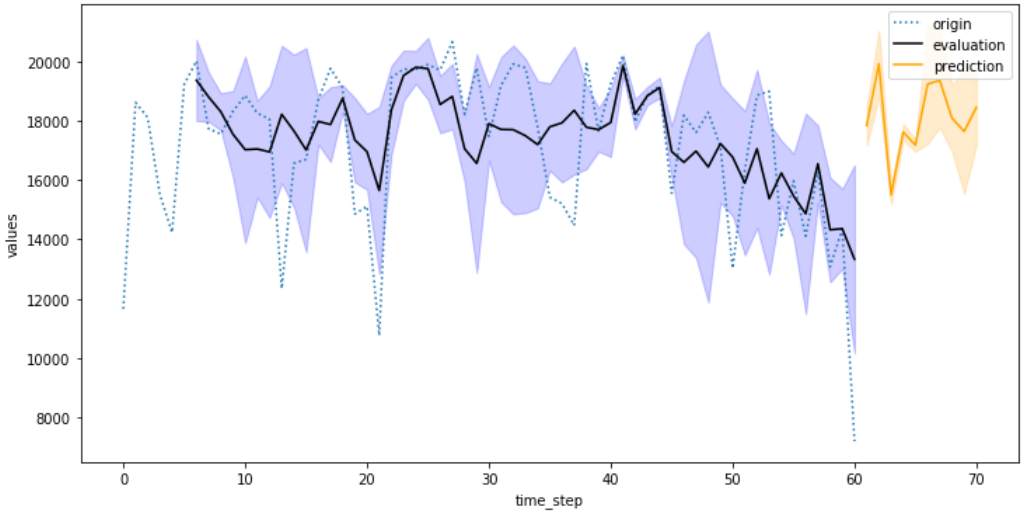
In [31]:
```
y
```

Out[31]:
```
array([11652, 18616, 18108, 15513, 14231, 19229, 20018, 17734, 17549,
       18316, 18856, 18256, 18047, 12338, 16564, 16695, 18705, 19768,
       19112, 14854, 15114, 10755, 19456, 19741, 19732, 19892, 19712,
       20663, 18225, 19782, 17470, 19192, 19916, 19790, 17783, 15419,
       15206, 14467, 19929, 17646, 19250, 20204, 17961, 18860, 19185,
       15556, 18183, 17607, 18280, 17123, 13021, 16341, 18858, 18993,
       14131, 15969, 14106, 16289, 13075, 14344,  7188], dtype=int64)
```

Dated data prediction

In [32]:
```
Tensorflowxy(x,y)
```

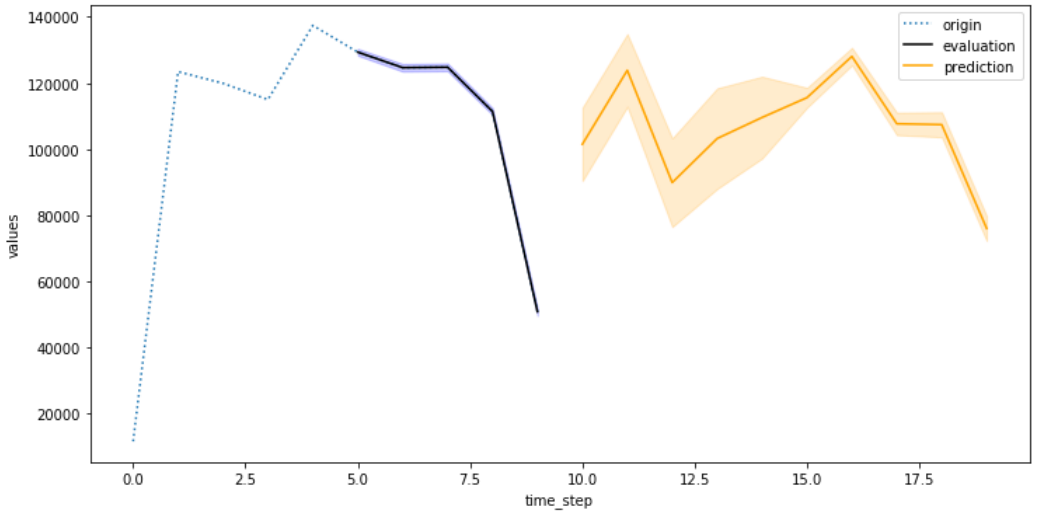
```
WARNING:tensorflow:Using temporary folder as model directory: C:\Users\chaet\AppData\Local\Temp\tmpe89fd2j9

[ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47
 48 49 50 51 52 53 54 55 56 57 58 59 60] [11652 18616 18108 15513 14231 19229 20018 17734 17549 18316 18856 18256
 18047 12338 16564 16695 18705 19768 19112 14854 15114 10755 19456 19741
 19732 19892 19712 20663 18225 19782 17470 19192 19916 19790 17783 15419
 15206 14467 19929 17646 19250 20204 17961 18860 19185 15556 18183 17607
 18280 17123 13021 16341 18858 18993 14131 15969 14106 16289 13075 14344
  7188]

WARNING:tensorflow:Skipping summary for covariance, must be a float, np.float32, np.int64, np.int32 or int or a seria
lized string of Summary.
WARNING:tensorflow:Skipping summary for mean, must be a float, np.float32, np.int64, np.int32 or int or a serialized
string of Summary.
WARNING:tensorflow:Skipping summary for observed, must be a float, np.float32, np.int64, np.int32 or int or a seriali
zed string of Summary.
WARNING:tensorflow:Skipping summary for start_tuple, must be a float, np.float32, np.int64, np.int32 or int or a seri
alized string of Summary.
WARNING:tensorflow:Skipping summary for times, must be a float, np.float32, np.int64, np.int32 or int or a serialized
string of Summary.
WARNING:tensorflow:Input graph does not use tf.data.Dataset or contain a QueueRunner. That means predict yields forev
er. This is probably a mistake.
```



Weekly data prediction

In [34]:
```
y1 = np.array(divvy_trips.groupby('Week_number')['trip_id'].count())
x1 = np.array(range(len(y1)))
x1 = pd.to_numeric(x1)
Tensorflowxy(x1,y1)
```

```
WARNING:tensorflow:Using temporary folder as model directory: C:\Users\chaet\AppData\Local\Temp\tmpqfzldx8t

[0 1 2 3 4 5 6 7 8 9] [ 11652 123449 119926 115003 137421 129352 124663 124794 111419  50896]

WARNING:tensorflow:Skipping summary for covariance, must be a float, np.float32, np.int64, np.int32 or int or a seria
lized string of Summary.
WARNING:tensorflow:Skipping summary for mean, must be a float, np.float32, np.int64, np.int32 or int or a serialized
string of Summary.
WARNING:tensorflow:Skipping summary for observed, must be a float, np.float32, np.int64, np.int32 or int or a seriali
zed string of Summary.
WARNING:tensorflow:Skipping summary for start_tuple, must be a float, np.float32, np.int64, np.int32 or int or a seri
alized string of Summary.
WARNING:tensorflow:Skipping summary for times, must be a float, np.float32, np.int64, np.int32 or int or a serialized
string of Summary.
WARNING:tensorflow:Input graph does not use tf.data.Dataset or contain a QueueRunner. That means predict yields forev
er. This is probably a mistake.
```



In [ ]: